Cognitive Radio Programming: Existing Solutions and Open Issues

Mickaël Dardaillon, Kevin Marquet, Jérôme Martin, Tanguy Risset, Henri-Pierre Charles

Cognitive Radio Programming: Existing Solutions and Open Issues

Mickaël Dardaillon, Kevin Marquet, Jérôme Martin, Tanguy Risset, Henri-Pierre Charles

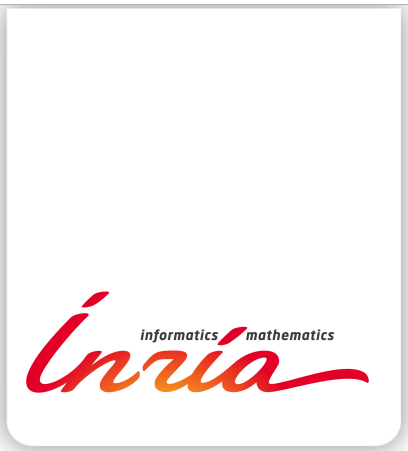
RESEARCH REPORT

N° 8358

September 2013

Project-Team Socrate

ISSN 0249-6399 ISRN INRIA/RR--8358--FR+ENG



Cognitive Radio Programming: Existing Solutions and Open Issues

Mickaël Dardaillon*, Kevin Marquet*, Jérôme Martin†, Tanguy Risset*, Henri-Pierre Charles‡

Project-Team Socrate

Research Report n° 8358 — September 2013 — 24 pages

Abstract: Software defined radio (SDR) technology has evolved rapidly and is now reaching market maturity, providing solutions for cognitive radio applications. Still, a lot of issues have yet to be studied. In this paper, we highlight the constraints imposed by recent radio protocols and we present current architectures and solutions for programming SDR. We also list the challenges to overcome in order to reach mastery of future cognitive radios systems.

Key-words: Software radio, Cognitive radio, Computer architecture, Reviews, Digital communications, Programming model

This work is partially supported by Région Rhône Alpes ADR 11 01302401.

* Université de Lyon, Inria, INSA-Lyon, CITI-Inria, F-69621, Villeurbanne, France
† CEA-Leti, Minatec campus F-38054, Grenoble, France
‡ CEA-List, CTL 7, av de Palestine, F-38610 Gières France

RESEARCH CENTRE GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Programmer la radio cognitive : solutions existantes et problèmes ouverts

Résumé : La radio logicielle a évolué rapidement pour atteindre la maturité nécessaire pour être mise sur le marché, offrant de nouvelles solutions pour les applications de radio cognitive. Cependant, beaucoup de problèmes restent à étudier. Dans ce papier, nous présentons les contraintes imposées par les nouveaux protocoles radios, les architectures matérielles existantes ainsi que les solutions pour les programmer. De plus, nous listons les difficultés à surmonter pour maitriser les futurs systèmes de radio cognitive.

Mots-clés : Radio logicielle, Radio cognitive, Architecture matérielle, Communication numérique, Modèle de programmation

1 Introduction

Radio technologies have been developed in a static paradigm: protocols, radio resources allocation and access network architectures were defined beforehand, providing non-adaptable radio systems. Nowadays, the saturation of radio frequency bands calls new era of radio networking which will be characterized by self-adaptive mechanisms. These mechanisms will rely on *software radio technologies*.

The concept of software radio has been coined by J. Mitola in his seminal work during the early 90's [56]. While implementing the whole radio node in software is still an utopia, many architectures now hitting the market include some degree of programmability.

With emerging Software-Defined Radio (SDR) technology, many questions arise that are related to the software layer of a software radio machine: How will this kind of platform be programmed? How can we write programs which are portable from one terminal to another? To answer these questions, programmers have to know how the architectural characteristics of SDR systems can be abstracted so as to provide portable code. Unfortunately, there is no agreement on the hardware architecture embedded in a mobile terminal with SDR facilities. Various technologies are used: ASIC, FPGA, DSP, GPP, etc. These technologies are often mixed and sometimes the term configurable is more adequate than programmable for them.

Studying simultaneously architectures, programming environments and programming models for emerging SDR systems is of crucial importance because of the need to define the hardware abstraction layer of SDR systems: the *radio hardware abstraction layer* (R-HAL).

In 2010, two important surveys where published [58, 72]. In [72], Tore Ulversøy provides a very complete review of SDR challenges related to software architecture, computational requirements, security, certification and business for SDR systems. Some SDR architecture prototypes are mentioned but are not the main topic of the study, and many other prototypes have been delivered since 2010. In [58], Palkovic et al. provide a precise comparative study between the Imec Bear Platform and other important SDR multi-core architectures. The comparison is made for architectures and programming flows. Our study extends to the study of programming models for waveform description languages and includes more recent SDR platforms.

We first provide an up-to-date review of existing SDR hardware platforms classifying them into five categories. Programming models and programming tools used in these platforms are not mature yet, most of these platforms being currently programmed "by hand". No common language, format or API has emerged yet, hence it is impossible to compare precisely the performance of the different approaches. All performance results presented in this paper are taken from bibliography.

We illustrate, with LTE as an example, the problems appearing with modern digital telecommunication physical layer protocols: fast terminal reconfiguration, data-dependent data flow. We also give an insight of what should be used as programming model for the programming of SDR platforms.

The rest of the paper is organized as follows: section 2 provides a brief summary of radio and SDR technology, we also present the example extracted from the LTE protocol that illustrates the difficulties of SDR programming. Section 3 presents the survey of hardware SDR platforms, categorizes and provides synthetic performance comparisons between them including power consumption when available. In section 4, we focus on programming environments for SDR and more precisely on the problem of defining a language for describing waveforms, i.e. physical layer of radio communication protocols. Section 5 reviews the remaining most important open problems: defining a programming model for SDR and a Hardware Abstraction Layer for SDR.

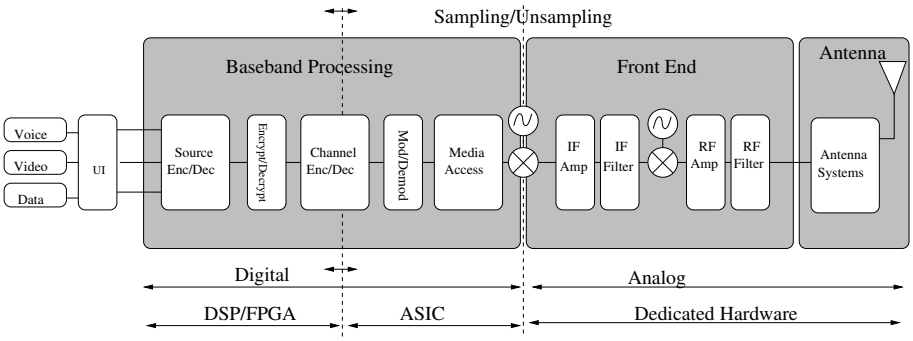


Figure 1: Radio Block Diagram, highlighting separation between digital and analog parts, as well as programmable, configurable and fixed hardware parts.

2 Cognitive Radio Technology

2.1 SDR technology

The different components of a radio system are illustrated in Fig. 1. Of course, all of the digital components may not be programmable, but the bigger the programmable part (DSP/FPGA part on Fig. 1), the more *software* the radio. Dedicated circuits are usually needed, for which the term configurable is more adapted than programmable. In a typical SDR, the analog part is limited to two frequency translations to an intermediate band and down to the base band which is sampled, and all the signal processing is done digitally.

To encourage a common meaning for the term "SDR", the SDR Forum (recently renamed Wireless Innovation Forum) proposes to distinguish five tiers. Tier 0 corresponds to hardware radio, Tier 1 corresponds to software controlled radio (only the control functions are implemented in software) and Tier 2 corresponds to software-defined radio and is the most popular definition of SDR: the radio includes software control of modulation, bandwidth, frequency range and frequency bands. Tier 3 and 4 are not realistic today.

Building an SDR terminal includes choosing a computing platform for the digital part, a sampling frequency and a radio front-end. In addition to the careful choice of a computing platform, the designer must make a trade-off between sampling frequency and terminal complexity. For instance, sampling a signal at 4.9 GHz (hence with a 10 GHz sample rate) is today not available with reasonable power consumption. Even with an evolution to lower power ADC, a high bandwidth ADC would produce more samples, hence the Front End characteristics (bandwidth, ADC resolution, etc.) constrains the digital part in terms of computing power. In this paper, we focus on the digital part represented on the left side of Fig. 1, assuming an adequate Front End is available for the platform.

The hardware platforms we review in the following are considered from a programmer point of view. They target the implementation of wireless communication protocol stacks from application down to physical layer (including baseband processing and intermediate frequency conversion), for emission (TX) and/or reception (RX).

2.2 Cognitive Radio

A *Cognitive Radio* is a wireless communication system that can sense the air, and decide to configure itself in a given mode. Tier 2 SDR platforms are natural candidates for cognitive radio implementation but cognitive radios do not have to be SDR.

The main feature enabled by spectrum sensing capacity is called *dynamic spectrum management*: the system is able to configure radio-system parameters in an autonomous manner. These radio-system parameters include transmission power, frequency band, modulation, channel and source coding, but might as well include higher level parameters such as waveform (physical layer protocol), MAC protocol, routing protocol and other networking characteristics. In that case, the term "autonomous" means "without human decision", i.e. automatic. However in many cases the decision cannot be taken independently of neighbouring communicating devices implied in the communication. This leads to the new scientific field of *distributed algorithms for radio resource allocation*.

Distributed algorithms are used when the decision of choosing a coding scheme or a frequency band has to be shared by many radio terminals. This perspective opens many new research problems and many new applications at the same time. For instance, distributed algorithms can be used to optimize interference cancellation globally, hence optimizing power consumption. Another example is the use of *relay*, i.e. transmission of packets from neighbour to neighbour according to routing decisions done at the physical layer, as opposed to routing decisions taken at a higher level in the protocol stack. Relay can be used for reducing transition power or to improve quality of transmission using network coding techniques.

From the research point of view, distributed algorithms open new fields: complexity and optimality of distributed solution to dynamic spectrum management. In point to point communication, OFDM techniques are approaching theoretical optimal bound for spectrum efficiency, that is the information rate that can be transmitted over a given bandwidth. But if cooperation between terminals is allowed, theoretical bounds are much more difficult to compute and technologically there is place for large improvements in communication systems that use cognition, cooperation and distributed decision algorithms. These problems are tackled in a new scientific field named *network information theory*.

2.3 Example of LTE

In this section we will show why recent radio communication protocols require a specific attention from a programmer's perspective. These protocols introduce harder real-time and dynamicity constraints that make usual computation models inefficient. Historically, structured programming led to imperative programming followed by fruitful evolutions for general purpose programming: object-oriented programming, functional languages, threads etc. Simultaneously, domain-specific programming models have been adopted in many fields, the most well known being reactive programming model for real-time control, and dataflow programming model for signal processing.

Dataflow programming model has been popularized by the *dataflow domain* of Ptolemy [30], implementing Khan's process networks [46]. Dataflow programming assumes that the flow of data is statically known and that the executed computation does not depend on data values. This condition has been verified for fifty years of signal processing but is not satisfied anymore by, for instance, LTE protocol. A more complete analysis of the existing computation models for SDR is given in section 4, below we simply give an example of the practical problems encountered when programming LTE on an SDR platform.

LTE is a mobile communication standard, developed by the 3GPP (3rd Generation Partnership Project) and approved into ITU (International Telecommunications Union). Fig. 2 represents the global flow for decoding a LTE frame (release 8, mode 5), from the mobile equipment point of view. A LTE frame is composed of 10 sub-frames of 1ms each, each sub-frame being composed of 14 symbols in time and 2048 sub-carriers in frequency for a 20 MHz bandwidth. The transmission uses MIMO technology (Multiple-Input and Multiple-Output), with up to 4 antennas for transmission on the base station and 2 antennas for reception on the mobile equipment.

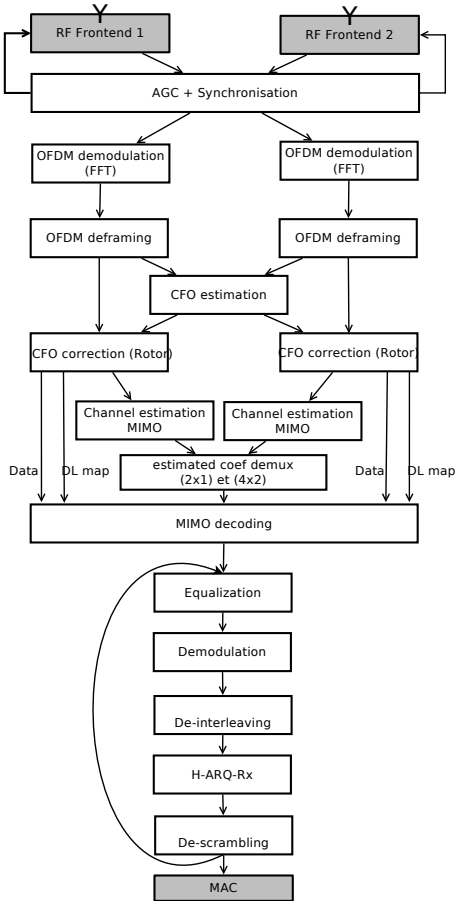


Figure 2: LTE pipeline flow, CFO correction and PDCCH decoding requires specific attention.

In this flow, we pay attention to the carrier frequency offset (CFO) correction. This correction is performed by analyzing specific resource elements of the frame called reference signals or *pilots*. Pilots are *within* the sub-frame that is corrected, hence the estimation of the CFO (CFO-estimation on Fig. 2) must be performed very quickly, and, more challenging, the carrier frequency offset correction (CFO-correction on Fig. 2) must be configured using the result of the CFO estimation. Hence this reconfiguration is dynamic and real-time, it should occurs in less than 100μs (10% of the computation time for a sub-frame). This is the first problem that SDR platform designers (and programmers as well) encounter: how to design a system that can reconfigure so quickly.

The second problem occurs after the FFT and MIMO decoding has brought samples in the frequency domain. The sub-frame is then composed of a matrix of symbols, not all of them being addressed to a given user, because LTE encodes several users in the same sub-frame. Extracting the symbols for a given user requires decoding the Physical Downlink Control CHannel (PDCCH). However, PDCCH format is encoded within the sub-frame itself in the Physical Control Format Indicator CHannel (PCFICH) (see Fig. 3), which must therefore be decoded beforehand. Depending on the PDCCH a certain number of symbols in the PDSCH (Physical Downlink Shared CHannel) will have to be sent to the rest of the flow (for demodulation etc.). Moreover, the LTE standard states that the modulation scheme (QPSK or 16QAM for instance), should also be encoded in the

PDCCH. Fig. 3 illustrates the symbol matrix in the frame, the successive decoding of PCFICH, PDCCH and PDSCH are illustrated in Fig. 2 by the back arrow after de-scrambling.

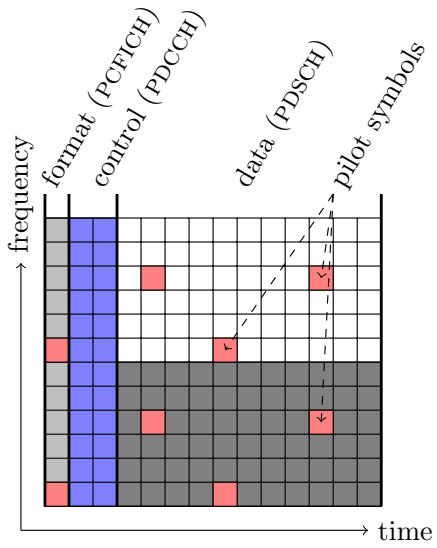


Figure 3: Resource allocation in an LTE PRB (Physical Resource Block).

This is the second problem that we highlight which definitely cannot be expressed in a static dataflow programming model: the number of data to be transmitted, as well as the computation to perform on each piece of data, are dependent on the data themselves. This is one of the main motivations of the work presented here: how to express such a computation in a language that is generic enough to be compiled on various SDR platforms?

3 Survey of Hardware Platforms for SDR

In order to classify the SDR platforms, we need to define objective criteria. Trying to define criteria based on used technology can be tricky, as most platforms are heterogeneous. Moreover, the technology used may not be a relevant criterion for platform users. The user will mainly be interested in the three following features: programmability and computing power, which will condition the supported protocols, as well as energy consumption which we believe will be the limiting factor for technology adoption. Choosing a computing platform for a given application is a trade-off between these features.

However, from the programmer point of view, the architecture is of major importance because it will have a crucial impact on programming models and tools used on the platform. We end up with five categories of SDR architectures:

1. General-purpose CPU approach
2. Co-processor approach
3. Processor-centric approach
4. Configurable units approach

5. Programmable blocks approach

Each approach is described in its corresponding subsection, and examples of existing implementations are given.

3.1 General-purpose CPU approach

The general-purpose CPU approach (depicted in Fig. 4) uses a general purpose computer processor to provide a computing platform. It offers a flexible and easy way to program the platform, but with a high energy consumption for a performance objective.

With the CMOS technology continuous evolution, one could imagine that future computers will be able to compute all protocols in real-time. However, as shown in [72], the increase in data throughput is higher than the increase in computing power. Therefore, this kind of architecture will only be able to support past protocols, unless it can make use of higher parallelism.

USRP The Universal Software Radio Peripheral (USRP) [9] is representative of the General-purpose CPU approach. It is composed of high frequency ADC/DAC which sample the signal in intermediate frequency. A FPGA converts and stores baseband signal. Most of the signal processing is done by a CPU connected to the FPGA by a USB link (USRP1) or an Ethernet link (USRP2). The platform is widespread and supported by third party software. It is aimed to work with GNU radio, but is also compatible with National Instruments' LabView and Mathworks' Matlab.

Quicksilver The Quicksilver [7] module is similar in behaviour with the USRP. However, it is only able to receive RF signals.

Microsoft SORA Recently, Microsoft developed SORA [69]. This platform is connected to the computer by a PCIe bus, which permits low latency and high throughput data transmission. It makes extensive use of modern CPU features to perform 802.11b/g processing in real-time.

3.2 Co-processor approach

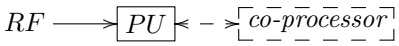


Figure 4: General-purpose CPU approach with optional co-processor

In order to accelerate the signal processing, optimizations of the general-purpose CPU approach have been explored recently. As depicted in Fig. 4, they rely on the addition of a co-processor to perform heavy processing. It reduces the price to pay in terms of energy while keeping high programmability and flexibility.

The work presented in [40] uses a GPU as a co-processor in a GNU radio flow. It permits gains of a factor 3 to 4 in processing speed.

KUAR The Kansas University Agile Radio (KUAR) [55] uses an embedded PC associated to a FPGA. The choice of the model of computation is left to the programmer, ranging from a full VHDL implementation (category described in subsection 3.5) to a full processor implementation close to the GNU radio flow.

Other developments use generic DSP as central processor, which provides higher efficiency while keeping high programmability.

Texas Instruments Texas instruments offers a three-core DSP with specialized symbol and chip rate accelerators. This product provides programming flexibility for WCDMA base cells, with support for up to 64 users and different protocols [11].

Imec ADRES The ADRES (Architecture for Dynamically Reconfigurable Embedded Systems) [18] developed by Imec is a coarse-grain reconfigurable architecture. It is built around a main CPU and the ADRES accelerator. The ADRES is seen by the processor as a VLIW co-processor, while being an array of 16 functional units. Each one is an SIMD processor, which leverages data parallelism. The processor is programmed using the DRESC compiler [54], in ANSI `C`. The DRESC compiler generates code to unroll loops and compute them using the ADRES accelerator. It targets telecommunications with benchmarks on 802.11n up to 108 Mbps and LTE up to 18 Mbps, and an average consumption of 333 mW [18].

Hiveflex Hiveflex [2] produces accelerators based on many small cores. These accelerators are scalable in terms of number of cores, depending on the application. All wireless protocols are targeted, from 802.11 to LTE, but no details about computing power or energy consumption are given. The accelerators are sold as soft IPs within HiveCC, the company SDK.

3.3 Processor-centric approach

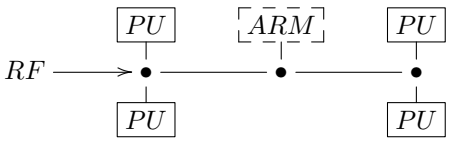


Figure 5: Processor-centric approach with optional central control processor.

Previous architectures offer only limited task parallelism. The next categories fill this gap using tailored architectures with heterogeneous types of processors. One approach to get efficient and specialized platforms is to use dedicated processors. In this approach, dedicated processors are used to compute signal processing. Both central and distributed control are considered in this section.

The processor-centric approach has a high programmability, but the flexibility of the platform is reduced by its specific architecture. The architecture concept is depicted on Fig. 5.

NXP EVP16 The NXP EVP16 [15], presented in 2005, is composed of several computing units. An ARM processor provides control and LINK/MAC layers. A conventional DSP, a vector processor and several hardware accelerators are used for signal processing. The vector processor is built as a vectorized pipeline and addressed as a VLIW. It performs UMTS for a 640 kbps throughput at 35 MHz, with a maximum of 300 MHz [15].

Infineon MuSIC Infineon built the MuSIC [61] as a multi-DSP solution for SDR. The control is done by an ARM processor. Signal computation is processed by 4 SIMD DSP and dedicated processors for filtering and channel encoding. Power consumption in WCDMA mode is 382 mW for

the worst case and 280 mW for a typical case. This chip is provided as a commercial solution under the name X-GOLD SDR 20 by Infineon [62]. It is programmed using a mix of C code and assembly code for critical processing.

Sandblaster The Sandblaster architecture [66] is built around 3 entities: the fetch and branch unit, the integer and load/store unit, and the SIMD vector unit. Task parallelism is managed by a Token Triggered Threading (T^3) component, which provides hardware support for multithreading. On the SB3011 [33], 4 sandblaster cores are integrated and controlled by an ARM processor. It is programmed in ANSI C with a dedicated compiler. Maximum consumption is 171 mW for WCDMA at 384 kbps [33].

University of Michigan ARDBEG The University of Michigan at Ann Arbor developed the SODA [74] SDR platform, and its prototype version ARDBEG [75]. SODA was developed as a complete software SDR solution. It consists of an ARM for control and 4 SIMD DSPs for signal processing. ARDBEG builds on that platform by adding a hardware turbo decoder and optimizing DSPs for signal processing. All programming is made using C code. Consumption results on ARDBEG for WCDMA and 802.11a are under 500 mW [75].

University of Dresden Tomahawk The University of Dresden, Germany, developed the Tomahawk SDR chip [49], aiming at LTE and WiMAX. It uses two Tensilica RISC processors for control, six vector DSPs and two scalar DSPs for signal processing, as well as ASIC accelerators for filtering and decoding. The scheduling is done by dedicated hardware and C code is used for programming. No protocol has been implemented yet on this platform. From the authors estimation, the platform consumption is about 1.5 W [49].

Picochip Picochip [60] approaches signal processing using many small cores. These cores are mapped on a deterministic matrix. A C-based development tool flow is provided by the company. No benchmark is provided for this chip. However, the company is announcing OFDM and 4G base stations as reference applications on its website.

UC Davis AsAP The University of California at Davis developed the Asynchronous Array of Simple Processors [71] (AsAP). This project aims at providing signal processing computation using small processors. All processors can communicate with their nearest neighbours, in a grid-like array. Version 2 adds hardware accelerators for FFT, Viterbi and video motion estimation, while increasing the total number of cores to 167. Complete 802.11a/g is processed at 54 Mbps using 198 mW [71].

3.4 Configurable units approach

In order to offer lower energy consumption, some platforms substitute DSP for configurable units. The difference between specialized DSPs and configurable units is very thin : a DSP is able to process any computation, whereas a configurable unit is too specialized to do so. This implies a big difference in term of programmability: to gain more performance, the DSP flexibility is abandoned in favor of configurable units. This leads to platforms which are much more difficult to program.

Fujitsu SDR LSI Fujitsu developed the SDR LSI [63] in 2005. The platform makes extensive use of hardware accelerators, associated to reconfigurable processors. All these components are connected to a crossbar data network, and controlled by a central ARM processor. The chip was able to run 802.11a/b with a maximum throughput of 43 Mbps [63].

Imec BEAR The BEAR SDR platform [59] is the evolution of the ADRES from Imec. It is constituted of an ARM processor for control and three ASIPs for coarse time synchronization on different front ends. Two ADRES coarse-grain configurable architectures, as described in subsection 3.2, are used for baseband processing with a Viterbi accelerator. The platform can be programmed with `C` or Matlab code, using the Imec development chain. In terms of energy consumption, BEAR achieves 2×2 MIMO OFDM at 108 Mbps for 231 mW [28]. Imec is licensing the BEAR platform as an IP block.

CEA-Leti Magali The Magali SDR chip [24] is developed by the CEA-Leti as a telecommunication demonstration platform. It is built on a Network-on-Chip, each peripheral having an access to the network, with an ARM processor controlling configurations. Computation is done by coarse-grain reconfigurable cores called Mephisto and reconfigurable IPs for OFDM, decoding and deinterleaving. Smart memory engines [53] are distributed on the Network-on-Chip and act like DMAs, while also providing data rearrangement capabilities. The chip performs 4×2 MIMO LTE reception in the most demanding scenario with a consumption of 236 mW [44].

CEA-Leti Genepy CEA-Leti Genepy [44] is using a larger granularity for its distributed approach. It is based on Magali [24] technology, using the Network-on-Chip and the coarse-grain configurable cores. The control carried out by the ARM processor is undertaken by distributed small RISC processors. Each cell on the network is composed of two Mephisto cores, one Smart Memory Engine and a RISC controller. The platform is purely homogeneous, with no hardware accelerators. In terms of computing power, 4×2 MIMO LTE reception is processed with a total consumption of 192 mW [44].

EURECOM ExpressMIMO The ExpressMIMO is developed as a configurable units approach on a FPGA by EURECOM [57]. All the configurable units share a common network interface, DMA engine and microcontroller, and each has a specific configurable IP for data processing. The board targets OFDM MIMO implementations and uses the OpenAirInterface open-source framework [5]. A more recent implementation should be available soon [65].

University of Twente Annabelle University of Twente, Netherlands, developed the Annabelle SDR chip. It is also built on a Network-on-Chip, using coarse-grain reconfigurable cores. An ARM processor is used for control, and accelerator modules (Viterbi, etc.) are connected to the ARM through an AMBA bus. Only OFDM specific benchmarks have been published at the time of submission [76].

3.5 Programmable blocks approach

The last approach uses programmable blocks and is mainly constituted of FPGAs. It doesn't provide programmability as it is, but great flexibility to create tailored architectures. Programmable blocks offer high computing power for moderate energy consumption.

XiSystem The XiSystem [52] is a VLIW architecture featuring 3 concurrent datapaths, including a PiCoGA (Pipelined Configurable Gate Array). The PiCoGA is an oriented datapath FPGA which executes specific instructions for the processor at run-time. The development is made with `C` to provide code for both the VLIW and the PiCoGA. It is aimed at embedded signal processing in general, with a benchmark on MPEG2 encoding and an average consumption of 300 mW [52].

Rice University WARP The Rice University has developed WARP [10], an open SDR platform. The computation is done by a Xilinx Virtex FPGA. Programming uses VHDL language. An open source community is led by the Rice University to offer open source implementations on the platform. For instance, it contains a MIMO OFDM Reference Design that can be extended based on Xilinx XPS tool.

Rutgers University WINC2R WINC2R is an original platform for SDR developed by the Rutgers University. The platform is built on a FPGA, with softcore processors and accelerators. Softcore processors can be programmed with GNU radio. Computation flow can be balanced on processors or accelerators, depending on the constraints. Moreover, by using an FPGA, accelerators can be chosen and tuned during development. 802.11a has been implemented on the platform [64].

Lyrtech The Lyrtech company [4] offers development tools and platforms for SDR based on FPGA. Development is done using Simulink model-based approach. The platform is presented as supporting MIMO WiMAX. Many other companies offer similar products based on FPGA [8, 6].

	Availability	Application	Prog.	Cons.
USRP [9]	commercial	N/A	`C++`	$\approx$ PC
TI C64+ [11]	commercial	base station	`C/ASM`	6000 mW
MuSIC [61]	commercial	WCDMA	`C/ASM`	$\leq$ 382 mW
Sandblaster [66]	IP licence	WCDMA	`C`	171 mW
ARDBEG [75]	prototype	WCDMA	`C`	$\leq$ 500 mW
BEAR [59]	IP licence	MIMO OFDM	matlab/`C`	231 mW
Magali [24]	prototype	MIMO OFDM	`C/ASM`	236 mW
ExpressMIMO[57]	prototype	MIMO OFDM	`C`	N/A
WARP [10]	commercial	MIMO OFDM	VHDL	N/A
Lyrtech [4]	commercial	N/A	matlab/VHDL	N/A
ASAP [71]	prototype	802.11a/g	N/A	198 mW
Genepy [44]	prototype	MIMO OFDM	`C/ASM`	192 mW

Table 1: Comparison of key SDR platforms based on the published performance results

3.6 Analysis

In order to better understand each category, we summarize the main characteristics for key platforms that use different approaches in Table 1. Energy consumption is not defined for FPGA-based platforms because it is heavily dependent on the configuration. Based on these key platforms, we draw trends on the application fields of each category.

If you don't want to study energy consumption nor architecture algorithm adequacy, the general-purpose CPU approach is the easiest way to go. However, if you intend to study energy consumption or computing power impact, this approach is not recommended. Indeed, dedicated hardware platforms have very different behaviours compared to generic processors. This makes

it difficult to establish a relationship between computing power and energy consumption for the generic approach and others. As an example, for a given protocol, computing requirements in terms of number of operations per second may vary with a factor of 100 in the literature, depending on the architecture granularity.

In order to study computing power and to have the lowest energy consumption, a heterogeneous approach which exploits hardware acceleration is a better starting point. In this family, using DSPs as in Imec's solution [59] or configurable blocks as in Magali [24] seems a pragmatic and efficient approach, these platforms being dedicated, and hence optimized, for SDR.

Unfortunately, using such a solution makes you heavily dependent on the platform architecture, and porting a waveform to a different architecture can be tricky. Providing a common HAL is a real challenging but promising way to develop practical multi-platform SDR.

Alternatively, the programmable blocks approach provides a flexible and efficient platform for prototyping thanks to the large adoption of FPGA technology. It can be versatile in the architecture choice, see the radically different approaches from [10] and [64] for example.

The most obvious conclusion from this SDR architecture survey is that no common architecture model could be extracted to provide, as it is the case for the general purpose *processor*, a *hardware abstraction layer* that could be used to help programming cognitive radio applications. We are now going to study the efforts that have been made to provide a programming environment adapted to cognitive radio.

4 Cognitive radio programming framework survey

As we have seen before, there has been a lot of efforts to set up dedicated SDR hardware. From these works, we can conclude that *i*) hardware support is necessary to match performances and low-power requirements of modern radio protocols and *ii*) it is not feasible to write traditional `C/ASM` code and map it manually anymore.

Programming and executing waveforms is clearly an application scope of the general problem of programming parallel machines, and this has to be taken into account when programming an SDR hardware. We now review research efforts that have been made to program SDR platforms efficiently. We first give in section 4.1 an insight of some programming environments used to program *more than one* SDR architecture. Then we focus on one central aspect of radio programming: *waveform programming*. Expressing a waveform, i.e. the physical part of the radio communication protocol, in a high-level language is a challenge. We classify the radio programming environments according to the programming model they use to express waveforms in sections 4.2, 4.3 and 4.4.

4.1 Cognitive Radio programming environment

There are two distinct important issues to address in the programming environment. The first one is the programming model used to specify the waveform (described in the following section), and the second is the global programming framework that will enable this programming model to be efficiently implemented on most of the platforms mentioned in section 3. Choosing the right programming framework is not a simple matter of comparing objectively pros and cons, it highly depends on strategic choices in companies and cultural acceptance by programmers.

The Software Communication Architecture (SCA) applicative framework was launched by the US department of defense within the Joint Tactical Radio System project (JTRS). It is an example of *top-down designed* framework. SCA re-uses major technologies coming from distributed software programming such as CORBA (Common Object Request Broker Architecture) for instance. The SCA has been implemented in OSSIE [34] and in military devices too. The OSSIE set of tools of the

SCA framework is an initiative from the American department of defense intending to provide a graphical environment for rapid prototyping of waveforms. It allows connection of components and generation of the corresponding code. However, the SCA framework is probably doomed to failure as the department of defense cancelled the project after it failed the Army's Network Integrated Environment testing [3].

Relying on the success of open-source software development, the GNUradio project [1] proposes to implement software-defined radio systems using a library of signal processing blocks written in `C++` for performance-critical parts, with `Python` programming language to interface these blocks. Initially dedicated to the Universal Software Radio Peripheral (USRP [9]) SDR hardware from Ettus Research, it recently received attention from many other hardware providers. However, this approach is currently implemented in general purpose CPU platforms and will encounter timing problems when complex MIMO OFDM protocols will have to be implemented. Some implementations, as for instance the OpenAir Interface [5], use real-time OS such as RT-Linux to improve the quality of real-time signal processing handling.

Many dedicated environments are based on a graphical interface coupled with dedicated IPs, as for instance Simulink coupled with Mathworks tools to program FPGAs or LabView. Recent trends based on OpenMP [23] or OpenCL [43] are emerging [73], but have not gain enough attention yet.

4.2 Imperative concurrent waveform programming

For an embedded software programmer, the easiest way to program an SDR platform is to use an imperative languages (generally `C` language) associated with threads to express parallelism. It has been used to program waveforms for both heterogeneous and homogeneous parallel platforms. For instance, the different units of the BEAR SDR platform [59] are programmed using `C` and Matlab code.

The efficient programming and execution of waveforms is tightly coupled with advances in the programming techniques for heterogeneous platforms. Although not yet evaluated for waveform programming, the ExoCHI [73] programming environment and the Merge [51] framework (based on ExoCHI) are proposals aiming at easing the programming of heterogeneous platforms while achieving good performances. The proposed solution is to extend OpenMP with intrinsic functions and dynamically map the software on available resources.

Cohen *et al.* [25] propose a similar approach in which programs are compiled into a specific bytecode and then compiled dynamically to the different accelerators available on the platform. This approach has not been evaluated on SDR platforms yet.

Many isolated works concentrate on the use of hardware accelerators. The Dresc [54] compiler allows to unroll loops in order to execute parallelized code on a specific accelerator made of 64 functional units.

The integration of the GPU in a SDR programming model has also been studied. Horrein *et al.* compare [41] different system architectures for using the GPU for SDR programming. Their work is based on OpenCL [38] and GNUradio.

4.3 Dataflow waveform programming

Numerous research works present arguments in favor of a paradigm shift and propose to program waveforms using dataflow languages. These languages relies on a *Model of Computation* (MoC) where a program is represented as a directed graph $G = (V, E)$. An actor $v \in V$ represents a computational module or a hierarchically nested subgraph. A directed edge $e \in E$ represents a FIFO buffer from its source actor S to its destination actor D. Dataflow graphs follow a data-driven

execution: an actor v can be executed (fired) only when enough data samples are available on its input edges. When firing, v consumes a certain amount of samples from its input edges and produces a certain number of samples on its output edges.

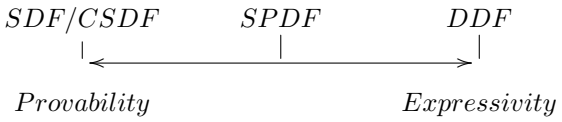


Figure 6: Representation of the balance between provability and expressivity in dataflow computation models.

Many dataflow-compliant programming models have been proposed for specific applications; they are illustrated in Fig. 6. *Synchronous DataFlow* (SDF) means that the number of tokens necessary for an actor to fire is known at compile-time. In this case, static scheduling of actors can be performed and the size of the buffers between actors can be bounded. In *Dynamic DataFlow*, data samples consumed and produced by an actor at each firing can vary dynamically at runtime, and can even be 0 in order to provide more flexibility for programming. As a drawback, theoretical analysis capabilities are reduced. Between synchronous and dynamic dataflow formalisms, a wide amount of models have been proposed, e.g. *Cyclo-Static Dataflow* (CSDF) [16], Schedulable Parametric DataFlow (SPDF) [32]. The goal was to look for a trade-off between the ability to statically analyze programs and the expressivity of the languages. For instance, using SDF to model a LTE waveform will lead to over-estimate the necessary resources at runtime because dynamic behaviour shown in section 2.3 will not be captured.

StreamIt [70] is a programming language that allows to describe programs in an SDF manner, through the use of *filters* and *split-join* operators. It comes with tools able to perform static analyses and optimization's of the dataflow graph. The compiler can generate `C` code for threads, that the programmer has to map manually on the available hardware resources. The underlying CSDF MoC is restricted to a single flow, which makes StreamIt not usable for complex and dynamic waveforms such as LTE.

ΣC [37] is a proposal to program waveforms using an extension of `C`. The corresponding MoC is more expressive than SDF thanks to non-deterministic extensions but still allows some static analyses to be performed such as bounding memory usage. However it does not allow dynamic behavior of actors, which is a limiting approach when attempting to describe waveforms such as LTE. The experimental platform used for ΣC is a many-core processor, and the tools allowing to compile and map for it are not freely available.

Past works have demonstrated the interest of programming using a general purpose language augmented with some primitives that allow to build the dataflow graph. Following the *Stream Virtual Machine* [47] approach, StreamWare [39] proposes to write dataflow graphs in a dedicated `C` API and schedule them at runtime on top of a general purpose processor. The same approach was applied to LTE [14] using a virtual machine (LUA). The waveform program contains dedicated reconfiguration primitives written in LUA language and interpreted directly on a controller. Those works do not restrict to a particular dataflow MoC.

In a similar approach, the Nucleus tool flow [22] comprises a set of tools able to compile and map waveforms. It uses the MAPS [21] framework in order to describe actors (so-called *nuclei*) in the CPN language. Different implementations can be provided for each actor, and a user-guided mapping computes a scheduling.

The non-open tool *SystemVue* [29] allows to model waveforms in SDF or TSDF (*Timed* SDF) form. It was used as a basis for a recent work [42] attempting to address the dynamic behavior of LTE by introducing vectorizers and serializers in the dataflow graph.

The DiplodocusDF approach [35] extends UML profiles to model dataflow applications. Thanks to a formal semantic, the resulting dedicated UML language can be simulated. Code for the underlying hardware can also be generated, but the mapping has to be done manually.

4.4 Mixing programming paradigms

The SPEX approach [50] proposes to program waveforms using three paradigms. *Kernel* SPEX allows a sequential, C-style imperative programming that can be useful for SIMD or VLIW compilation. *Stream* SPEX can be used to program using the dataflow paradigm, following the KPN MoC. *Synchronous* SPEX relies on the paradigm used in synchronous languages such as Esterel or Signal. The distribution of the paradigms is left to the programmer but all parts are included in a C++ program in which 1) the choice of the paradigm is indicated by a keyword, 2) no dynamic object creation is allowed. The compilation of this program involves one compiler for each paradigm.

In a similar manner, IRIS [68] proposes to write SPHY and FPHY engines. SPHY implements SDF components while FPHY implements KPN components. The mapping of the engines is left to the programmer. The framework provides support for reconfiguration: components may trigger a signal which will lead to the reconfiguration of the kernels.

Lime [12] is a Java-based language with extensions to express more parallelism. In Lime, the same method body can be used as a standard function or as an actor in order to program in a dataflow style. In this case, Lime also provides a *match* operator allowing actors to execute at different rates to communicate, thus extending SDF while keeping analysis capabilities. It is associated with a compilation/execution that generates Java bytecode, C, or Verilog, in order to be able to choose between different implementations for each actor.

Finally, It is worth mentioning that many research teams have been working on designing complete system from high level specification in the so-called *hardware-software co-design* domain. These works brought advances in specific aspects such as platform based design or high level synthesis tools such as CatapultC for instance. Although these works did not led to a dedicated SDR environment but might, in the near future, lead to refinement-based SDR programming environment.

4.5 Discussion

The survey of SDR programming environments provided above shows that, as it was the case for hardware architectures, there is no agreement on what should be a programming environment for cognitive radio. However, there is a clear trend toward a paradigm shift in order to handle protocols such as LTE (see section 2.3). These new protocols are very different from previous signal processing applications, that can be programmed with static traditional parallelization techniques (SDF and/or traditional compilation techniques).

The arguments in favor of dataflow programming models for SDR are:

- Radio waveforms are inherently dataflow because they operate on large data sequences. Although not infinite — they are grouped into frames — radio waveforms still require static (software or hardware) filters that are easily expressed through dataflow actors.
- Sofware-defined radio applications require huge computation performances and hence need to efficiently use parallelism available in hardware. Dataflow formalisms allow for better parallel implementation because it naturally exposes parallelism in many ways: task parallelism, data parallelism and pipeline structure of the program.
- Dataflow programs have a restricted expressivity that allows them to be analyzed in order to verify some properties such as the absence of deadlock, or to improve timing analyses.

Such analyses are important since waveforms are becoming more and more complex. Hence, by hand analyses will become impossible. New analysis tools will be needed to ensure properties on these programs.

Although they introduce a paradigm shift, dataflow approaches seem necessary. We believe that, at least mixed approaches between this paradigm and imperative concurrent languages will succeed in providing a compromise between programmability, performance and provability. Section 5 reports on open issues we have identified concerning the programming of SDR platforms, and reviews different basic research tracks to address them.

5 Open Issues

In previous section we have seen that, in order to address the challenges of new communication protocols such as LTE, many works are based on dataflow computation models and dataflow programming languages. However, there is a gap between these works and experimental prototypes. We now report on issues to be addressed in order to fill this gap. We have identified two main directions in which technology should be improved: mapping flows and hardware abstractions.

5.1 Mapping flows

One open problem with existing programming frameworks is that they all require a manual *mapping* of the application onto the SDR architecture. The mapping is the phase where the initial specification is split into blocks that are assigned to the different IPs of the architecture. We review below some recent works that attempt to take into account waveform characteristics in SDR programming languages and provide tools to improve the mapping flow.

5.1.1 Handling dynamicity

Recent works propose new dataflow MoCs that take into account dynamic adaptations required by new communication protocols. One problem with such solutions is to provide languages and compilers for such MoCs. Although there have been many advances in this field [45], providing such tools requires an important research and development effort. An example of a new dataflow MoC is Schedulable Parametric DataFlow [32], language in which it is possible to change actors' parameters while still allowing static analyses.

New compilers such as ORCC [36] provides support for dataflow programming and dynamic dataflow using just-in-time (JIT) compilation to modify dataflow at runtime, targeting CPUs. Other work [27] considers dynamical reconfiguration on heterogeneous platforms based on FPGAs.

5.1.2 High-level data structures

Another issue in the portability of SDR applications is the ability to directly handle high-level data structures. Indeed, in section 2, we saw that the LTE protocol operates on vectors and matrices. However, current dataflow MoCs and languages only allow manipulation of token flows without making high-level data structures apparent. This prevents compilers and execution layers from: *i*) optimizing the placement of data, and *ii*) taking into account the specifics of communication features (DMA, Network-on-Chip, interrupts etc.). The same issue has been reported in non telecom application fields [70].

MVDF [42] makes one step in this direction by proposing to write dynamic vectorization actors able to produce vectors from a dynamic number of tokens. Other works that could be used here include:

- ArrayOL [19] is a vector-based specification model that allows expression of static transformations of multidimensional arrays.
- Block Parallel [17] is a proposal to specify data organization in *input* and *output* actors. In order to ease compilation, it is limited to two-dimensional arrays.
- OpenCL [38] and Brook [20] include specific functions for operations on multidimensional arrays.
- Slices [26] is a language dedicated to multidimensional data reorganization, with associated tools that statically map programs onto parallel platforms.
- The Sequoia [31] programming language allows programmers to explicitly divide programs into data movement and computation steps in order to optimize data placement at runtime. Computation units can only share or move data via a parent memory node in the memory hierarchy.

These works propose static solutions that do not take into account the dynamic variation of data type and/or size. Manipulating dynamic high-level dataflow structures remains an unsolved problem today.

5.2 HAL for SDR

The problem of a common *hardware abstraction layer* (HAL) for SDR is definitely not solved. An idea that is emerging slowly is that an Application Programming Interface (API) should be standardized for SDR hardware platforms. This API should include for example an FFT function with various parameters, and probably high-level telecom-specific functions such as Viterbi or Turbo encoding and decoding. However the precise specification of this API has not been done yet.

Related to this issue, some works attempt to abstract specific hardware in order to lower the need for manual adaptation of the mapping flow. As described in the previous section, one common approach is to consider the use of a dataflow virtual machine [47, 14, 39]. These approaches do not address the problem of mapping waveforms onto the hardware. On the contrary, the Nucleus approach [22] and ExoCHI [73] are able to map computation units at runtime, but their mapping procedure cannot be easily extended to many hardware platforms.

Recently, The HDCRAM environnement was prototyped [48] by Moy et al. This environement will target dynamic reconfiguration of radio protocols on various plateform (DSP, FPGA). It has been used with GNUradio and still has to be tested in other environments.

5.3 Resource sharing

Another open problem when programming SDR and specifying waveforms, the behaviors of which depend on data contents at runtime, is to take into account, at the specification level, the concurrent execution of multiple waveforms on the same platform. Many hardware platforms [24, 65] include hardware mechanisms to ease this *radio context switch* but very few programming environments address this issue.

Siyoum et al. [67] show the interest of building different scenarios for a given waveform and express the relationship between each one at the MoC level. This allows static verification of timing properties and optimization of resource usage at runtime. This approach is limited to scenarios written in SDF form.

5.4 Discussion

An illustrative example of the difficulty of providing a programming environment portable to different hardware platforms, as are today's retargetable compilers, is given by the Magali chip [24]. This chip, dedicated to 4G Telecommunication applications contains an OFDM IP which performs FFT as well as deframing (suppression of the band guard). Hence, a mapping tool should be able to gather the software block for FFT and deframing and to map them onto the OFDM IP: there is not necessarily a one-to-one correspondence between actors and hardware IPs.

A way to reach portability is to agree on a single API for programming SDR applications. Current solutions are far from this goal: each hardware platform comes with its own specific abstraction.

New MoCs, detailed in section 5.1.1, have improved analysis capabilities but are currently not considered in actual design flows. Hence we lack information concerning the performances of these new models once compiled and executed.

One way to bridge the gap between defining new, high-level, analyzable models and providing enhanced execution layers is to statically compute some information and properties on the programs, and use them at runtime to take accurate decisions. Such an approach is currently used by MAPS [21], but in a very limited manner since it only uses traces and hand-written information.

Another approach that seems promising to improve portability and performances is the dynamic compilation. The goal is to use a JIT compiler in order to compile dynamically code embedded in a high-level form such as a bytecode. The benefit from this approach is to take advantage of runtime information to compile and map more efficiently. Such an approach has been proposed by Cohen et al. [25]. The LiquidMetal [13] approach compiles code into Java bytecode and could therefore be a good starting point for experimenting.

6 Conclusion

In this paper we reviewed the cognitive radio technologies from hardware and software points of view. We started by illustrating new constraints introduced by protocols such as LTE and their impact on current programming models. We provided a review of the different categories of SDR platforms and their possible application fields, and we discussed the programming models used to program these platforms, with a current shift to a new dataflow programming paradigm. After these observations, we described open issues to bridge the gap between hardware and software, highlighting *i*) the need for new mapping flows to program SDR platforms efficiently and *ii*) the need for SDR HAL allowing software reuse from one SDR generation to another.

A promising research direction we are investigating at the moment is the design of this HAL to abstract from the different categories we have seen in this paper.

References

[1] Gnu radio framework. http://www.gnuradio.org.

[2] Hiveflex. http://www.siliconhive.com.

[3] The jtrs ground mobile radio failure. http://arstechnica.com/information-technology/2012/06/how-to-blow-6-billion-on-a-tech-project/.

[4] Lyrtech. http://www.lyrtech.com.

[5] Open air interface. http://www.openairinterface.org.

[6] Pentek. http://www.pentek.com.

[7] Quicksilver. http://www.philcovington.com/QuickSilver.

[8] Sundance. http://www.sundance.com.

[9] Universal software radio peripheral (usrp). http://www.ettus.com.

[10] Warp. http://warp.rice.edu.

[11] Sanjive Agarwala, Arjun Rajagopal, Anthony Hill, Mayur Joshi, Steven Mullinnix, Timothy Anderson, Raguram Damodaran, Lewis Nardini, Paul Wiley, Peter Groves, and Others. A 65nm C64x+ multi-core DSP platform for communications infrastructure. In *Solid-State Circuits Conference, ISSCC. Digest of Technical Papers. IEEE International*, pages 262–601. IEEE, 2007.

[12] Joshua Auerbach, David F. Bacon, Ioana Burcea, Perry Cheng, Stephen J. Fink, Rodric Rabbah, and Sunil Shukla. *A compiler and runtime for heterogeneous computing*, pages 271–276. Number 1. ACM Press, Jun 2012.

[13] Joshua Auerbach, David F. Bacon, Ioana Burcea, Perry Cheng, Stephen J. Fink, Rodric Rabbah, and Sunil Shukla. *A compiler and runtime for heterogeneous computing*, pages 271–276. Number 1. ACM Press, Jun 2012.

[14] Riadh Ben Abdallah, Tanguy Risset, Antoine Fraboulet, and J. Martin. *Virtual Machine for Software Defined Radio: Evaluating the Software VM Approach*, volume 1970, page 1977. 2010.

[15] Kees Van Berkel, Frank Heinle, Patrick P. E. Meuwissen, Kees Moerman, and Matthias Weiss. Vector Processing as an Enabler for Software-Defined Radio in Handheld Devices. *EURASIP Journal on Advances in Signal Processing*, (16):2613–2625, 2005.

[16] Greet Bilsen, Marc Engels, Rudy Lauwreins, and Jean Peperstraete. Cyclo-static dataflow. *Signal Processing, IEEE Transactions on*, 44(2):397–408, 1996.

[17] David Black-Schaffer. *Block parallel programming for real-time applications on multi-core processors*. PhD thesis, Stanford, CA, USA, 2008.

[18] Bruno Bougard, Bjorn De Sutter, and D. Verkest. A coarse-grained array accelerator for software-defined radio. *IEEE Micro*, pages 41–50, 2008.

[19] Pierre Boulet. Array-OL Revisited, Multidimensional Intensive Signal Processing Specification. Rapport de recherche RR-6113, INRIA, 2007.

[20] Ian Buck, Tim Foley, Daniel Reiter Horn, Jeremy Sugerman, Kayvon Fatahalian, Mike Houston, and Pat Hanrahan. Brook for gpus : Stream computing on graphics hardware. *ACM Trans. Graph.*, 23(3):777–786, 2004.

[21] J. Castrillon, R. Leupers, and G. Ascheid. Maps: Mapping concurrent dataflow applications to heterogeneous mpsocs. *IEEE Transactions on Industrial Informatics*, X(X):1–19, 2011.

[22] Jeronimo Castrillon, Stefan Schürmans, Anastasia Stulova, Weihua Sheng, Torsten Kempf, Rainer Leupers, Gerd Ascheid, and Heinrich Meyr. Component-based waveform development: the nucleus tool flow for efficient and portable software defined radio. *Analog Integrated Circuits and Signal Processing*, 69(2-3):173–190, Jun 2011.

[23] Robit Chandra, Leonardo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, and Ramesh Menon. *Parallel programming in OpenMP*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[24] Fabien Clermidy, Romain Lemaire, Xavier Popon, Dimitri Ktenas, and Yvain Thonnart. An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application. In *Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pages 449–456, Patras, Greece, August 2009. IEEE.

[25] Albert Cohen and Erven Rohou. *Processor virtualization and split compilation for heterogeneous multicore embedded systems*, page 102. ACM Press, 2010.

[26] Pablo de Oliveira Castro, Stéphane Louise, and Denis Barthou. A multidimensional array slicing dsl for stream programming. In *Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems*, CISIS '10, pages 913–918, Washington, DC, USA, 2010. IEEE Computer Society.

[27] J.-P. Delahaye, J. Palicot, C. Moy, and Leray P. Partial reconfiguration of fpgas for dynamical reconfiguration of a software radio platform. In *Mobile and Wireless Communications Summit, 2007. 16th IST*, pages 1–5, Budapest, Hungary, July 2007.

[28] V Derudder, B Bougard, A Couvreur, A Dewilde, S Dupont, L Folens, L Hollevoet, F Naessens, D Novo, P Raghavan, and Others. A 200Mbps+ 2.14 nJ/b digital baseband multi processor system-on-chip for SDRs. In *VLSI Circuits, Symposium on*, pages 292–293, Kyoto, Japan, 2009. IEEE.

[29] Agilent EESof. Systemvue. http://www.home.agilent.com/en/pc-1297131/systemvue-electronic-system-level-esl-design-software.

[30] Johan Eker, Jörn W. Janneck, Edward A. Lee, Jie Liu, Xiaojun Liu, Jozsef Ludvig, Stephen Neuendorffer, Sonia Sachs, and Yuhong Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, Jan 2003.

[31] Kayvon Fatahalian, Daniel Reiter Horn, Timothy J. Knight, Larkhoon Leem, Mike Houston, Ji Young Park, Mattan Erez, Manman Ren, Alex Aiken, William J. Dally, and Pat Hanrahan. Sequoia: programming the memory hierarchy. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[32] Pascal Fradet, Alain Girault, and Peter Poplavko. *SPDF: A Schedulable Parametric Data-Flow MoC*, pages 769 – 774. Number 1. Mar 2012.

[33] John Glossner, Daniel Iancu, Mayan Moudgill, Gary Nacer, Sanjay Jinturkar, Stuart Stanley, and Michael Schulte. The Sandbridge SB3011 Platform. *EURASIP Journal on Embedded Systems*, pages 1–16, 2007.

[34] Carlos R. Aguayo González, Carl B. Dietrich, Shereef Sayed, Haris I. Volos, Joseph D. Gaeddert, P. Max Robert, Jeffrey H. Reed, and Frank E. Kragh. Open-source sca-based core framework and rapid development tools enable software-defined radio education and research. *Comm. Mag.*, 47:48–55, October 2009.

[35] Jair Gonzalez-Pina, Rabea Ameur-Boulifa, and Renaud Pacalet. Diplodocusdf, a domain-specific modelling language for software defined radio applications. *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 1–8, Sep 2012.

[36] J. Gorin, M. Wipliez, F. Prêteux, and Mickaël Raulet. A portable video tool library for mpeg reconfigurable video coding using llvm representation. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pages 183–190, Edinburgh, Scotland, October 2010.

[37] Thierry Goubier, Renaud Sirdey, Stéphane Louise, and Vincent David. ΣC: A programming model and language for embedded manycores. *Algorithms and Architectures for parallel processing*, pages 385–394, 2011.

[38] Khronos OpenCL Working Group. The opencl specification.

[39] Jayanth Gummaraju, Joel Coburn, Yoshio Turner, and Mendel Rosemblum. *Streamware: programming general-purpose multicore processors using streams*, pages 297–307. 2008.

[40] Pierre-Henri Horrein, Christine Hennebert, and Frédéric Pétrot. Adapting a SDR environment to GPU architectures. In *Wireless Innovation Forum (SDR Forum)*, Brussels, Belgium, June 2011.

[41] Pierre-Henri Horrein, Christine Hennebert, and Frédéric Pétrot. Integration of gpu computing in a software radio environment. *Journal of Signal Processing Systems*, 69(1):55–65, Dec 2011.

[42] Chia-Jui Hsu, José Luis Pino, and Fei-Jiang Hu. *A mixed-mode vector-based dataflow approach for modeling and simulating LTE physical layer*, pages 18–23. 2010.

[43] Pekka Jääskeläinen, Carlos S. de La Lama, Pablo Huerta, and Jarmo Takala. Opencl-based design methodology for application-specific processors. In Fadi J. Kurdahi and Jarmo Takala, editors, *ICSAMOS*, pages 223–230. IEEE, 2010.

[44] Camille Jalier, Didier Lattard, AA Jerraya, Gilles Sassatelli, Pascal Benoit, and Lionel Torres. Heterogeneous vs homogeneous MPSoC approaches for a mobile LTE modem. In *Conference on Design, Automation and Test in Europe*, pages 184–189, Dresden, Germany, March 2010.

[45] W.M. Johnston, JR Hanna, and R.J. Millar. Advances in dataflow programming languages. *ACM Computing Surveys (CSUR)*, 36(1):1–34, Mar 2004.

[46] G. Kahn. The semantics of a simple language for parallel programming. In J. L. Rosenfeld, editor, *Information processing*, pages 471–475, Stockholm, Sweden, Aug 1974. North Holland, Amsterdam.

[47] Francois Labonte, Peter Mattson, William Thies, Ian Buck, Christos Kozyrakis, and Mark Horowitz. *The stream virtual machine*, pages 267–277. IEEE Computer Society, Sep 2004.

[48] Oussama Lazrak, Pierre Leray, and Christophe Moy. HDCRAM Proof-of-Concept for Opportunistic Spectrum Access. In *Proceedings of the 15th Euromicro Conference on Digital System Design*, page 6 pages, Cesme, Izmir, Turkey, September 2012.

[49] Torsten Limberg, Markus Winter, Marcel Bimberg, Reimund Klemm, E. Matus, M.B.S. Tavares, Gerhard Fettweis, Hendrik Ahlendorf, and Pablo Robelly. A fully programmable 40 GOPS SDR single chip baseband for LTE/WiMAX terminals. In *Solid-State Circuits Conference, ESSCIRC. 34th European*, pages 466–469, Edinburgh, Scotland, September 2008. IEEE.

[50] Yuan Lin, Robert Mullenix, Mark Woh, Scott Mahlke, Trevor Mudge, Alastair Reid, and Krisztian Flautner. *SPEX: A programming language for software defined radio*, pages 13 – 17. Citeseer, Nov 2006.

[51] Michael D. Linderman, Jamison D. Collins, Hong Wang, and Teresa H. Y. Meng. *Merge : A Programming Model for Heterogeneous Multi-core Systems*, pages 287–296. Mar 2008.

[52] Andrea Lodi, Andrea Cappelli, Massimo Bocchi, Claudio Mucci, Massimilano Innocenti, C. De Bartolomeis, Luca Ciccarelli, Roberto Giansante, Antonio Deledda, Fabio Campi, and Others. XiSystem: a XiRisc-based SoC with reconfigurable IO module. *Solid-State Circuits, IEEE Journal of*, 41(1):85–96, 2006.

[53] Jérôme Martin, Christian Bernard, Fabien Clermidy, and Yves Durand. A Microprogrammable Memory Controller for High-Performance Dataflow Applications. In *European Solid-State Circuits Conference*, pages 348–351, Athens, Greece, September 2009. IEEE.

[54] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins. DRESC: A retargetable compiler for coarse-grained reconfigurable architectures. In *Field-Programmable Technology (FPT), IEEE International Conference on*, pages 166–173, 2002.

[55] Minden et al. KUAR: A Flexible Software-Defined Radio Development Platform. In *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, pages 428–439, Dublin, Ireland, April 2007. IEEE.

[56] J Mitola. Software Radios Survey, Critical Evaluation and Future Directions. In *Telesystems Conference, NTC. National*, pages 13/15–13/23, Washington, DC , USA, 1992. IEEE.

[57] Dominique Nussbaum, Karim Kalfallah, Christophe Moy, Amor Nafkha, Pierre Lerary, Julien Delorme, Jacques Palicot, Jérôme Martin, Fabien Clermidy, Bertrand Mercier, and Renaud Pacalet. Open Platform for Prototyping of Advanced Software Defined Radio and Cognitive Radio Techniques. In *12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pages 435–440, Patras, Greece, August 2009. IEEE.

[58] M. Palkovic, Praveen Raghavan, Min Li, A. Dejonghe, L. Van der Perre, and F. Catthoor. Future Software-Defined Radio Platforms and Mapping Flows. *Signal Processing Magazine, IEEE*, 27(2):22–33, 2010.

[59] M. Palkovic, Praveen Raghavan, Min Li, A. Dejonghe, L. Van der Perre, and F. Catthoor. Future Software-Defined Radio Platforms and Mapping Flows. *Signal Processing Magazine, IEEE*, 27(2):22–33, 2010.

[60] D. Pulley and R. Baines. Software defined baseband processing for 3G base stations. In *3G Mobile Communication Technologies, 3G. 4th International Conference on (Conf. Publ. No. 494)*, pages 123–127, London, UK, June 2003. IET.

[61] Ulrich Ramacher. Software-Defined Radio Prospects for Multistandard Mobile Phones. *Computer*, 40(10):62–69, 2007.

[62] U. Ramacher et al. Architecture and implementation of a Software-Defined Radio baseband processor. In *International Symposium on Circuits and Systems (ISCAS)*, pages 2193–2196, Rio de Janeiro, Brazil, May 2011.

[63] V.S.N.V.M. Saito and V.I. Sugiyama. Single-Chip Baseband Signal Processor for Software-Defined Radio. *FUJITSU Sci. Tech. J*, 42(2):240–247, 2006.

[64] S. Satarkar. *Performance analysis of the WiNC2R platform.* PhD thesis, 2009.

[65] Carina Schmidt Knorreck, Renaud Pacalet, Andreas Minwegen, Uwe Deidersen, Torsten Kempf, Raymond Knopp, and Gerd Ascheid. Flexible Front-End Processing for Software Defined Radio Applications using Application Specific Instruction-Set Processors. In *DASIP 2012, Conference on Design and Architectures for Signal and Image Processing, 23-25 October 2012, Karlsruhe, Germany*, Karlsruhe, GERMANY, 10 2012.

[66] Michael J. Schulte, John Glossner, Suman Mamidi, Mayan Moudgill, and S. Vassiliadis. A low-power multithreaded processor for baseband communication systems. *Computer Systems: Architectures, Modeling, and Simulation*, pages 333–346, 2004.

[67] Firew Siyoum, Marc Geilen, Orlando Moreira, Rick Nas, and Henk Corporaal. *Analyzing synchronous dataflow scenarios for dynamic software-defined radio applications*, pages 14–21. IEEE, Oct 2011.

[68] Paul D. Sutton, Jörg Lotze, Hicham Lahlou, Suhaib A. Fahmy, Keith E. Nolan, Barış Özgül, Thomas W. Rondeau, Juanjo Noguera, and Linda E. Doyle. Iris: an architecture for cognitive radio networking testbeds. *Communications*, (September):114–122, 2010.

[69] Kun Tan, He Liu, Jiansong Zhang, Yongguang Zhang, Ji Fang, and Geoffrey M. Voelker. Sora: high-performance software radio using general-purpose multi-core processors. *Communications of the ACM*, 54(1):99–107, 2011.

[70] W Thies. *Language and compiler support for stream programs.* PhD thesis, 2009.

[71] D.N. Truong, W.H. Cheng, T. Mohsenin, Zhiyi Yu, A.T. Jacobson, G. Landge, M.J. Meeuwsen, C. Watnik, A.T. Tran, Zhibin Xiao, E.W. Work, J.W. Webb, P.V. Mejia, and B.M. Baas. A 167-processor computational platform in 65 nm CMOS. *Solid-State Circuits, IEEE Journal of*, 44(4):123–127, 2009.

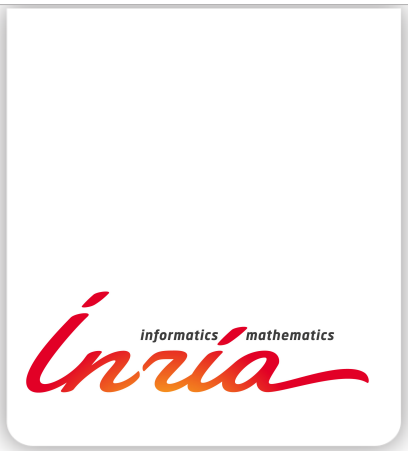
[72] Tore Ulversoy. Software defined radio: Challenges and opportunities. *Communications Surveys & Tutorials, IEEE*, 12(4):531–550, 2010.

[73] Perry H. Wang, Jamison D. Collins, Gautham N. Chinya, Hong Jiang, Xinmin Tian, Milind Girkar, Nick Y. Yang, Guei-Yuan Lueh, and Hong Wang. Exochi: architecture and programming environment for a heterogeneous multi-core multithreaded system. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '07, pages 156–166, New York, NY, USA, 2007. ACM.

[74] Mark Woh, Y. Harel, Scott Mahlke, T. Mudge, C. Chakrabarti, and K. Flautner. SODA: A Low-power Architecture For Software Radio. *33rd International Symposium on Computer Architecture (ISCA)*, pages 89–101, 2006.

[75] Mark Woh, Yuan Lin, Sangwon Seo, Scott Mahlke, Trevor Mudge, Chaitali Chakrabarti, Richard Bruce, Danny Kershaw, Alastair Reid, Mladen Wilder, and Others. From SODA to scotch: The evolution of a wireless baseband processor. In *Microarchitecture, MICRO. 41st IEEE/ACM International Symposium on*, pages 152–163, Como, Italy, November 2008. IEEE.

[76] Q. Zhang, a.B.J. Kokkeler, G.J.M. Smit, and K.H.G. Walters. Cognitive Radio baseband processing on a reconfigurable platform. *Physical Communication*, 2(1-2):33–46, March 2009.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399